# Intramural Sports Registration System (IMSS)

Peter Le (pal4ka)

Eugene Moy (em2ae)

CS4750 Project Report

## Introduction

A database system for an Intramural Sports Registration System is being modeled. This system will be based on the University of Virginia IM-Sports Model, albeit simplified. The main goal for the system, which will be referred to as “Intramural Sports System” or “IMSS” for short, is to facilitate the registration of athletes in teams that play in leagues and to keep track of statistics for games played.

The registration system should be able to set up players and teams to play in IM Sports games. Potential athletes should be able to register with certain personal information details to get into the IMSS. From there, they can form or join teams for certain IM leagues – for a certain sport. If a team meets the requirements of the league, they can be registered to play in the league in a series of games called a schedule. Each schedule has a certain time period in which the games are played as well as a registration deadline by which teams have to request to play in the games.

Not only should the IMSS register entities to set up games, but also make it easy to view statistics for players, teams, and results of games and schedules. This will allow users to size up their opponents, brag about their records, etc. Upcoming games, available leagues, available schedules within leagues to sign up for, and active teams should be easy to find.

Administrators for IMSS have permissions to make changes to game results and team stats. They should also have the ability to remove teams from leagues should they violate certain rules or regulations. Of course any of their changes should be reflected in the IMSS.

## Requirements

**Regarding Users**

- Users must have a unique computing ID (in theory, our system isn't linked to UVA or any other school) to register. This will be used to identify them.
- Users should be able to register via their computing ID and set up their own password. Their name, gender, and computing ID should not be able to change.
- Users' competition record for teams they are a part of should be kept track of. These records should be tied to results of games that they have played in the past.
- Users can be a member of multiple teams, but are limited to only one per league.

**Regarding Teams**

- All users should be able to set up a team that can be joined by appropriate users (for example a male wouldn't be able to join a team under a female volleyball league).
- The creator of the team should be designated as team captain.
- The competition records of each team should be recorded and updated when games' records are stored. Wins, losses, draws, and no contests will be counted by the admin.
- Teams should be able to register for play in certain leagues as long as they meet the requirements.
- When a captain drops from a team, the team disbands

**Leagues and Games**

- Leagues should fall under certain sports and at most one sport at a time.
- Each league has a schedule that has a certain start and end date and registration deadline by which teams have to register for to take part in the series of games.
- Each game's date, time, and location must be available.
- The results of each game in terms of what teams played and what each team scored should be kept track of. The result (win, loss, draw, or no contest) will be recorded as well.
- Results of each game should be reflected in team stats via admin changes
- Users should be able to view all leagues and past games played in those leagues regardless of eligibility to enter.

**Special Functionality**

- Each sport has a minimum number of players for teams. Only teams that meet this minimum player requirement may be included in the play schedule for a sports league.
- Rankings for each league should be shown through a view – the rankings are based on number of wins.

**Security**

- Only administrators should be able to manipulate game, schedule, league, and sport data.
- Passwords are encrypted with MD5
- Only administrators should be able to remove teams from leagues (disqualification) except for team deletions due to dropping.

**Exporting Data**

- Relevant table data should be able to be exported in JSON format.

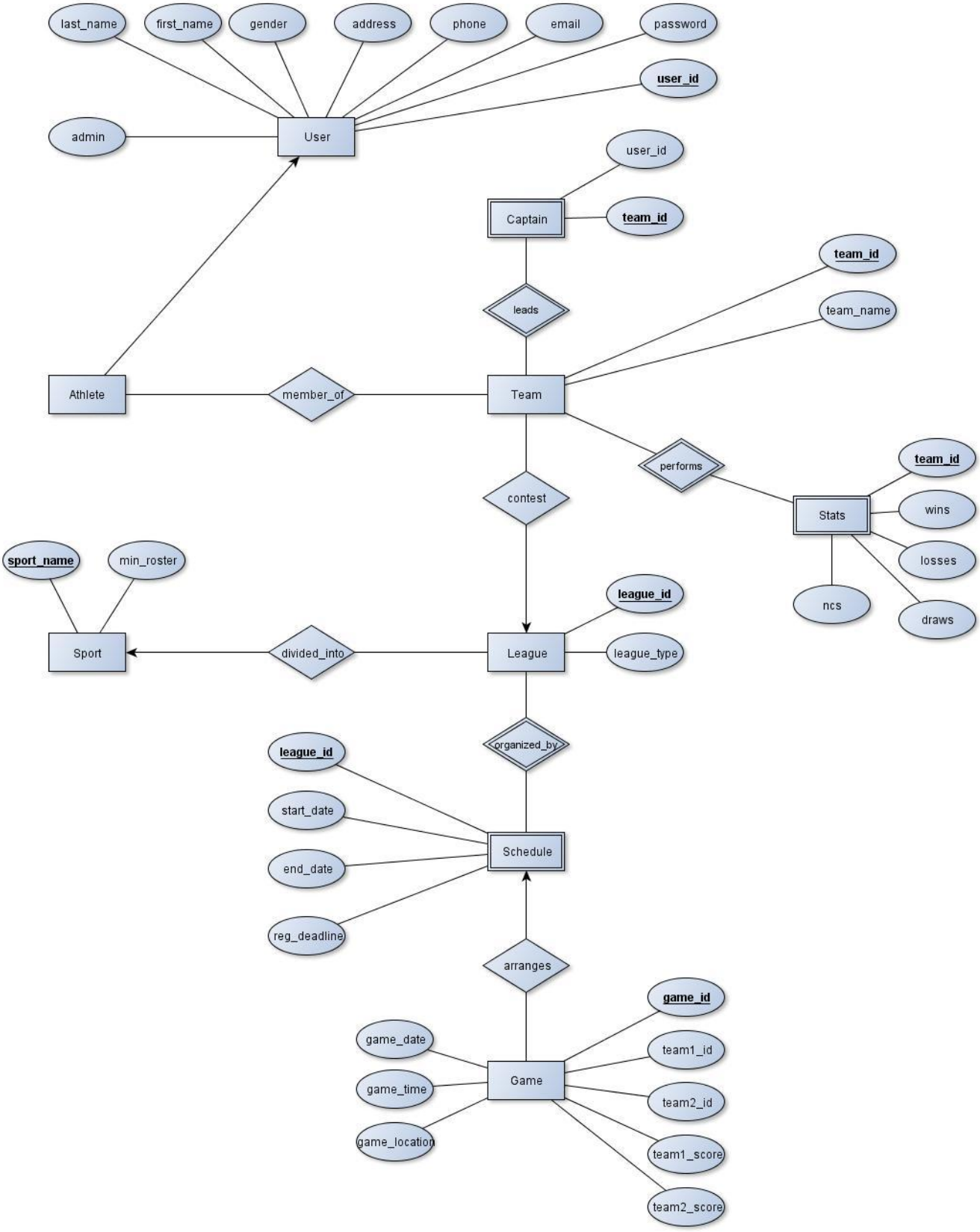
## Design Discussion

In designing the IMSS, we wanted to create something that would be simple to develop, use, and maintain. We decided an intramural sports website would be an ideal project to work with databases and web design because such a system involves gathering data across different categories (that we could put into different tables) with not much else. Essentially, a sports league site involves only relationships between entities, so we could focus on the database and web development side of things.

The IMSS is built on PHP. This is appropriate because a registration system for as large a community as a university should be online. An Eclipse project would not be as appropriate for such a system. In addition, PHP is relatively straightforward which eases the development process. AJAX is used for presentation and simplification (in terms of user friendliness).

The IMSS is designed to be secure on different levels. In order to protect user information at the application level, there is a distinction between non-users, users, and admins. Admins are marked with a variable in the database that determines whether or not upon login a user sees an admin interface. Non-users won't be able to see more than a page with login fields and a register link. This is to ensure that theoretical non-University students wouldn't be able to snoop around the website for names of students. Users can see such information, and ideally their user ID would be cross listed with X-University's records to see if they are a valid student. Users can register in teams and view leagues/games/results, but only admins can change non-user/team information. In addition to this aspect of security, passwords are encrypted with MD5 hashing. POSTs are used instead of GETs to hide information from the user, and procedures are used where appropriate. On the physical level we could not do much, and can only hope UVA is doing a good job of guarding its computer systems.

## ER Diagram


last_name
first_name
gender
address
phone
email
password
user_id
admin
User
user_id
Captain
team_id
team_id
leads
team_name
Athlete
member_of
Team
performs
team_id
contest
Stats
wins
losses
sport_name
min_roster
ncs
draws
league_id
Sport
divided_into
League
league_type
organized_by
league_id
start_date
Schedule
end_date
reg_deadline
arranges
game_id
game_date
team1_id
Game
game_time
team2_id
game_location
team1_score
team2_score

## Table Schema and 3NF Proofs

```
CREATE TABLE User (
                    user_id             varchar(7) NOT NULL UNIQUE,
                    last_name           varchar(20) NOT NULL,
                    first_name                  varchar(20) NOT NULL,
                    address             varchar(60),
                    phone                       varchar(10),
                    email                       varchar(30),
                    password varchar(32),
                    admin int(1),
                    gender int(1),
                    PRIMARY KEY (user_id)
                    )ENGINE=InnoDB;
```

R → (user_id, last_name, first_name, address, phone, email, admin, gender)
user_id → last_name, first_name, address, phone, email, admin
user_id is the primary key and others are dependent on it, so it is in 3NF

```
CREATE TABLE Team (
                    team_id             int(5) AUTO_INCREMENT UNIQUE,
                    team_name                   varchar(20) NOT NULL UNIQUE,
                    PRIMARY KEY (team_id)
                    ) ENGINE=InnoDB;
```

R → (team_id, team_name)
team_id → team_name
Two columns, in BCNF and 3NF

```
CREATE TABLE Captain (
                    team_id             int(5) UNIQUE,
                    captain             varchar(7)
                    PRIMARY KEY (team_id)
                    FOREIGN KEY (team_id) REFERENCES Team(team_id) ON DELETE CASCADE ON UPDATE CASCADE
                    FOREIGN KEY (captain) REFERENCES User(user_id)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE
                    ) ENGINE=InnoDB;
```

R → (team_id, captain)
team_id → captain
Two columns, in BCNF and 3NF

```
CREATE TABLE Stats (
                    team_id             int(5),
                    played                      int(2),

wins                        int(2),

                    losses                      int(2),
                    draws                       int(2),
                    ncs                         int(2),
                    PRIMARY KEY (team_id),
```

```
        FOREIGN KEY (team_id) REFERENCES Team(team_id) ON DELETE CASCADE ON UPDATE CASCADE
        )ENGINE=InnoDB;
```

R → (played, team_id, wins, losses, draws, ncs)
team_id → played, wins, losses, draws, ncs
team_id is the primary key, played, wins, losses, draws, ncs are not dependent on each other, in 3NF

```
CREATE TABLE League (
        league_id       int(4) AUTO_INCREMENT UNIQUE,
        league_type     varchar(10) NOT NULL,
        sport_name              varchar(20) NOT NULL,
        PRIMARY KEY (league_id, sport_name),
        FOREIGN KEY (sport_name) REFERENCES Sport(sport_name) ON DELETE CASCADE ON UPDATE CASCADE
        ) ENGINE=InnoDB;
```

R → (league_id, league_type, sport_name)
league_id → league_type, sport_name
In 3NF, league_type and sport_name aren't dependent , league_id is the primary key

```
CREATE TABLE Sport (
        sport_name              varchar(20) NOT NULL UNIQUE,
        min_roster              int(2) NOT NULL,
        PRIMARY KEY (sport_name)
        )ENGINE=InnoDB;
```

sport_name → min_roster
2 columns, sport_name is the primary key, this is in BCNF and therefore 3NF as well

```
CREATE TABLE Schedule (
        league_id       int(4),
        start_date      DATE,
        end_date        DATE,
        reg_deadline    DATE,
        PRIMARY KEY (league_id, start_date),
        FOREIGN KEY (league_id) REFERENCES League(league_id) ON DELETE CASCADE ON UPDATE CASCADE
        ) ENGINE=InnoDB;
```

R → (league_id, start_date, end_date, reg_deadline)
league_id, start_date → end_date, reg_deadline
Each non prime attribute is dependent on the primary key, nothing is dependent on each other, this is in 3NF

```
CREATE TABLE Game (
        game_id         int(6) UNIQUE AUTO_INCREMENT,
        game_date               DATE,
        game_time               TIME,
        game_location   varchar(20),
        team1_id        int(5),
        team2_id        int(5),
        team1_score     int(3),
        team2_score     int(3),
        PRIMARY KEY (game_id)
        ) ENGINE=InnoDB;
```

R → (game_id, game_date, game_time, game_location, team1_id, team2_id, team1_score, team2_score)

game_id → game_date, game_time, game_location, team1_id, team2_id, team1_score, team2_score
None of the non-prime attributes are dependent on each other. This is in 3NF.

```
CREATE TABLE member_of (
                user_id         varchar(7),
                team_id         int(5),
                PRIMARY KEY (user_id, team_id),
                FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE ON UPDATE CASCADE,
                FOREIGN KEY (team_id) REFERENCES Team(team_id) ON DELETE CASCADE ON UPDATE CASCADE
                ) ENGINE=InnoDB;
```

R → (user_id, team_id)
user_id → team_id
2 attributes, automatically in BCNF and 3NF

```
CREATE TABLE contest (
                team_id         int(5),
                league_id       int(4),
                PRIMARY KEY (team_id),
                FOREIGN KEY (team_id) REFERENCES Team(team_id) ON DELETE CASCADE ON UPDATE CASCADE,
                FOREIGN KEY (league_id) REFERENCES League(league_id) ON DELETE CASCADE ON UPDATE CASCADE
                )ENGINE=InnoDB;
```

R → (team_id, league_id)
team_id → league_id
2 attributes, automatically in BCNF and 3NF

```
CREATE TABLE divided_into (
                sport_name              varchar(20),
                league_id       int(4),
                PRIMARY KEY (league_id),
                FOREIGN KEY (league_id) REFERENCES League(league_id) ON DELETE CASCADE ON UPDATE CASCADE,
                FOREIGN KEY (sport_name) REFERENCES Sport(sport_name) ON DELETE CASCADE ON UPDATE CASCADE
                ) ENGINE=InnoDB;
```

R → (sport_name, league_id)
sport_name → league_id
2 attributes, automatically in BCNF and 3NF

```
CREATE TABLE arranges (
                league_id       int(4),
                game_id         int(6),
                PRIMARY KEY (game_id),
                FOREIGN KEY (game_id) REFERENCES Game(game_id) ON DELETE CASCADE ON UPDATE CASCADE,
                FOREIGN KEY (league_id) REFERENCES League(league_id) ON DELETE CASCADE ON UPDATE CASCADE
                ) ENGINE=InnoDB;
```

R → (league_id, game_id)
league_id → game_id
2 attributes, automatically in BCNF and 3NF

## Views

- Leaders show the champions of a league

```
CREATE VIEW leaders AS
SELECT sport_name, league_type, league_id, end_date, team_name, wins, losses, draws, ncs
FROM (Team NATURAL JOIN Stats NATURAL JOIN League NATURAL JOIN contest NATURAL JOIN Schedule)
WHERE wins=(SELECT MAX(wins) FROM (Team NATURAL JOIN Stats NATURAL JOIN League NATURAL JOIN Schedule))
```

Note: All tables are in InnoDB, and deletions and updates are cascading.

## MySQL Procedures

TeamRoster

```
DROP PROCEDURE IF EXISTS TeamRoster;

DELIMITER //
CREATE PROCEDURE TeamRoster(IN teamName VARCHAR(20))
BEGIN
        SELECT user_id, first_name, last_name
        FROM User NATURAL JOIN member_of NATURAL JOIN Team
        WHERE team_name = teamName;
END //
DELIMITER ;
```

TeamRoster('team_name') returns the roster of a team
i.e. CALL TeamRoster('Sweet Boys');

LeagueTeams('leagueType', 'sportName') returns list of teams in a specific league

```
DROP PROCEDURE IF EXISTS LeagueTeams;
DELIMITER //
CREATE PROCEDURE LeagueTeams(IN leagueType VARCHAR(20), IN sportName VARCHAR(20))
BEGIN
        SELECT team_name
        FROM Team NATURAL JOIN contest NATURAL JOIN League
        WHERE sport_name = sportName and league_type = leagueType;
END //
DELIMITER ;
```

i.e. CALL LeagueTeams('CoRec', 'Volleyball');

createTeam('teamID', 'userID', 'leagueID') create a team and set up

```
DROP PROCEDURE IF EXISTS createTeam;
DELIMITER //
CREATE PROCEDURE createTeam(IN teamID INT(5), IN userID VARCHAR(7), IN leagueID INT(4))
BEGIN
        INSERT INTO Captain(team_id, captain) VALUES (teamID, userID );
        INSERT INTO member_of(user_id, team_id) VALUES (userID, teamID );
        INSERT INTO Stats(team_id, played, wins, losses, draws, ncs) VALUES (teamID, '0', '0',  '0', '0', '0');
        INSERT INTO contest(team_id, league_id) VALUES (teamID, leagueID);
END //
DELIMITER ;
```

i.e. CALL createTeam('5', 'Team Awesome', 'Mark Sherriff', '231')

## Testing Procedures

In testing the IMSS we focused on making sure the requirements held, and that actions and their effects were natural to the users. For example when registering a team many tables had to be updated. A team has to be associated with a league, a game schedule, a captain, and its members have to be associated with the team. Whenever we inserted something like a team for example, we made sure that things matched up. This was more easily done by displaying data in PHP that involved data across tables – so when they were displayed we could easily cross-check. Another thing we tested for was deletion of

things that were also involved in other tables – possible leftover data. For example if a team is deleted, naturally the entries in other tables involving that team should also change. We tested the cascading of updates/deletes for anything that was a foreign key in another table. Going through the requirements and trying to do the contrary through the website was the quickest way to test for correct functionality.

As for testing security we mainly had to deal with what pages a type of user (non-registered, registered, or admin) could view at some time. So we went through all our pages as each type of user to see if our protocols were correct. In this way nobody stumble on a sensitive page.

## Sample Data & Queries

- Getting games that fall in a league and displaying team attributes

```
$sql="SELECT game_id, game_date, game_time, game_location, Team.team_name as team1_name,
X.team_name as team2_name, team1_id, team2_id, team1_score, team2_score

                FROM `arranges`

                NATURAL JOIN Game

                NATURAL JOIN contest NATURAL JOIN League

                NATURAL JOIN Team, Team AS X

                WHERE Team.team_id = Game.team1_id

                AND X.team_id = Game.team2_id AND league_type='".$leaguetype."'
AND sport_name='".$sport."'";
```

Where the league is Mens' Basketball..

| Game ID | Date | Time | Location | Team 1 | Team 1 Score | Team 2 Score | Team 2 |
|---|---|---|---|---|---|---|---|
| 7 | 2012-04-02 | 15:00:00 | AFC Court 1 | Sweet Boys | 21 | 11 | Monkey Do |
| 9 | 2012-04-09 | 15:00:00 | AFC Court 1 | Sweet Boys | 21 | 32 | Young Jedi Knights |
| 10 | 2012-04-09 | 15:00:00 | AFC Court 2 | Monkey Do | 12 | 30 | Hoo Tang Clan |
| 11 | 2012-04-16 | 15:00:00 | AFC Court 1 | Monkey Do | 19 | 24 | Young Jedi Knights |
| 8 | 2012-04-02 | 15:00:00 | AFC Court 2 | Young Jedi Knights | 24 | 20 | Hoo Tang Clan |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | 2012-04-16 | 15:00:00 | AFC Court 2 | Hoo Tang Clan | 27 | 17 | Sweet Boys |

- Deleting a league

```
$sql = "DELETE FROM League WHERE league_id='".$lid."' AND league_type='".$lt."' AND
sport_name='".$sport."';";
```

- Inserting a user into the database

```
$q = "SELECT * FROM ".$table_name." WHERE user_id='".$userid."'";
        $result = mysql_query($q) or die(mysql_error().": $q"); //this query will be used to check for
the same username
        $n1 = mysql_num_rows($result); //this will return 0 if there are no $table_name with that
name
        if($n1 != 0) { //if there are any $table_name, it will return how many there are, and we can stop
the signup
          die("<p><font color='red'>Error: The username is already in use.</font></p>"); //let them
know
        }
        $q = "SELECT * FROM ".$table_name." WHERE email='".$email."'";
                                $result = mysql_query($q) or die(mysql_error().": $q"); //this query will
check if the email is in use.
        $n2 = mysql_num_rows($result); //if you read above, you should understand what this is for.
        if($n2 != 0) { //someone else has that email
          die("<p><font color='red'>Error: The email address is in use by another user.</font></p>");
//so tell the user
        }

        $pass = md5($pass1,$md5c); //md5 the pass with the code entered at the top of the page
       $sql="INSERT INTO $table_name (user_id, last_name, first_name, address, phone, email,
password, gender, admin) VALUES('".$userid."', '".$lastname."', '".$firstname."', '".$address."',
'".$phone."', '".$email."', '".$pass."', '".$gender."', '0')";
```

## Data Printout

See other documents